Alternus Vera: Fake News Detection

Anvitha Karanam
Jahnavi Rangu
Leela Alekhya Vedhula
Manisha Yacham

Abstract
Fake News is any news that is either factually wrong, misrepresents the facts, and that spreads virally (or maybe to a targeted audience).What truly differentiates Fake News from simple hoaxes is the fact that it carefully mimics the "style" and "patterns" that real news usually follows. That's what makes it so hard to distinguish for the untrained human eye. It has increased enormously with the rise of the data-driven era which is not a coincidence when we consider the sheer volume of data that is being generated every second. It is almost ubiquitously used on every social media platform like Facebook, Twitter, WhatsApp, etc. and has become synonymous with the menace in society and politics in recent years. Hence, Fake news has now become a major concern in our society.

With so many advances in Natural Language Processing and machine learning, scope of identifying fake news has improved drastically. This paper presents various techniques used to identify the factors influencing fake news detection and the process of classifying the fakeness factor by using various NLP techniques.We chose Clickbait, Stance Detection, Spam Classification and BERT multi-class classification for detecting fake news.

I. INTRODUCTION

BERT Multi-Class Classification

Transfer learning models like Allen AI's ELMO, OpenAI's Open-GPT, and Google's BERT allowed many researchers to outperform multiple baseline models with minimal task like fine-tuning and provided the rest of the NLP community with pretrained models that could easily be fine-tuned and implemented with less data and less compute time to produce state of the art results.

BERT (Bidirectional Encoder Representations from Transformers), which was released in late 2018, is a method of pretraining language representations used to create models that NLP practitioners can then download and use. We can either use these models to extract high quality language features from our text data, or fine-tune these models on a specific task (classification, entity recognition, question answering, etc.) with our own data to produce state of the art predictions. Considering the above, we decided to consider BERT for multi-class classification as one of the primary factors in classifying the fakeness of the news. Specifically, we will take the pre-trained BERT model, add an untrained layer of neurons on the end, and train the new model for our classification task.

This paper presents various approaches to classify the fake news. A distillation, amalgamation approach is incorporated to perform feature engineering and ensemble technique is used to combine the speaker's truth history features and build a fake news classifier.

B. Spam Classification

With the growth in technology, we also come across a lot of spam information in our day to day life through emails, news, social media and each one of these is a big source of spreading information. The spread of misinformation is known to cause extensive harm. And as most of the people these days are online clients, there are high chances of them being vulnerable to misinformation as they perceive all that they run over web-based networking media as reliable. Detecting truthfulness of information can save people from falling into traps. Spam classification has become a hot topic of study and helps to determine the veracity of the information. This paper proposes a machine learning model that is trained with a liar dataset amalgamated and enriched with two other datasets that are obtained from kaggle fake news and by scrapping from politifact websites. We have applied different classifiers such as Logistic Regression, Naive Bayes, Support vector machine and trained it using the classifier with highest accuracy.This dataset is trained and is used to test on any information can classify it as spam or ham. Our experimental result showed that SVM and Logistic regression classifiers outperformed the other algorithms.

C. Stance Detection

Social media has become one of the main channels for people to share their views and communicate with society. We can often detect and recognize from these statements whether the person is in favor, against or neutral towards a given subject. Stance is defined as the expression of the speaker's standpoint and judgement toward a given proposition. Stance detection plays a major role in analytical studies measuring public opinion on social media, particularly on political and social issues. Social issues such as abortion, climate change, and feminism have been heavily used as target topics for stance detection on social media. This paper explains the classification approach we have followed to identify the stance of news articles.

D. Clickbait

Clickbait is used to describe a type of hyperlink on a web page which seduces a user to click a link to continue reading a specific article. Clickbaits, in social media, are exaggerated headlines whose main motive is to mislead the reader to "click" on them. Clickbait is a term used to describe a deceiving web content that uses ambiguity to prompt the user into clicking a link. It aims to increase the number of online readers in order to generate more advertising revenue. Hence, We have implemented a model which achieves high performance in identification of clickbaits.

II IMPLEMENTATION DETAILS

A. BERT Multi-Class Classification:

Datasets

We have used the "Liar, Liar Pants on Fire" dataset and LIAR PLUS dataset. The datasets consist of manually labelled short text. This data can be used for fact checking. The data set consists of 3 files: train.csv, test.csv and validate.csv that can be used to run the NLP and ML lifecycle. The datasets have 3 CSV files: test, training and valid. Each file has 14 columns: the ID of the statement. the label, the statement, the subject(s)/news, the speaker, the speaker's job title, the state info, the party affiliation. Column 9-13 are the total credit history counts, including the current statement like barely true counts, false counts, half true counts, mostly true counts, pants on fire counts. The final column is the context (venue / location of the speech or statement).

Scraping and Enrichment

We have scraped around 800 pages of data from the fact checking website, Politifact.com and added this data to the existing dataset for data enrichment and to yield better train/test results.

What did I try and what worked?

As the first step, Since we'll be training a large neural network it's best to take advantage of the GPU provided by Google Colab, otherwise training will take a very long time and in order for torch to use the GPU, we need to identify and specify the GPU as the device. Next, we must install the transformers package from Hugging Face which gives us a pytorch interface for working with BERT. In addition to supporting a variety of different pre-trained transformer models, the library also includes pre-built modifications of these models, In this paper we will use the pretrained model called BertForSequenceClassification.

Data Preprocessing and Feature Engineering

Dealing with **missing data/value** is one of the most tricky but common parts of data cleaning. While many models can live with other problems of the data, most models don't accept missing data.In this paper, we drop the entire observation as long as it contains a missing value.

We have used wordcloud to visualize the datasets and the correlation between words and the similarities between them

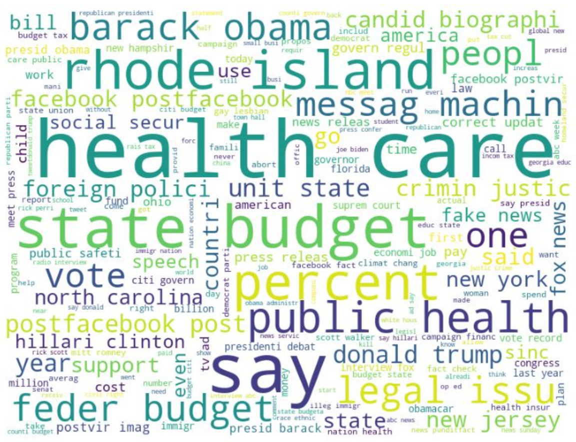


Figure 1. Word cloud visualization for train dataset

Another important aspect in training data is to make sure if the class distribution of the labels is even to yield better prediction results

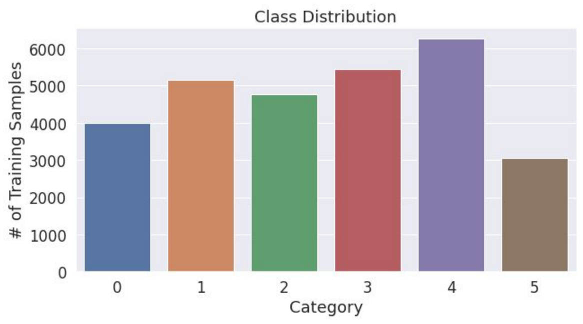


Figure 2. Visualization of even class distribution in the training dataset

As part of data cleaning we are also removing **Punctuations** as they are often unnecessary as it doesn't add value or meaning to the NLP model.

Stemming and **Lemmatization** is the process of reducing a word to its root form. The main purpose is to reduce variations of the same word, thereby reducing the corpus of words we include in the model. **Stop words** are irrelevant words that won't help in identifying a text as real or fake. We will use the "**NLTK**" library for stop-words. We are combining other features like speaker, source, subject along with the article headline and using this data to train the model to yield better predictions.

Distillation using LDA Topic Modeling

We have chosen to perform distillation using LDA topic modeling to add the top topics also as the feature input to the classification model as collecting well-balanced and carefully-assessed training data is important for developing robust fake news detection systems.

Tokenization

To feed our text to BERT, it must be split into tokens, and then these tokens must be mapped to their index in the tokenizer vocabulary. The tokenization must be performed by the tokenizer included with BERT. We'll be using the "uncased" version.

We must use the tokenizer provided by the model. This is because (1) the model has a specific, fixed vocabulary and (2) the BERT tokenizer has a particular way of handling out-of-vocabulary words.At the end of every sentence, we need to append the special [SEP] token.For classification tasks, we must prepend the special [CLS] token to the beginning of every sentence. The transformers library provides a helpful encode function which will handle most of the parsing and data prep steps for us.

The tokenizer.encode_plus function combines multiple steps for us:

1. Split the sentence into tokens.

2. Add the special [CLS] and [SEP] tokens.

3. Map the tokens to their IDs.

4. Pad or truncate all sentences to the same length.

5. Create the attention masks which explicitly differentiate real tokens from [PAD] tokens.

Here we have also added the encoded truth history counts to enhance the model classification results.

Sentence length and attention mask

BERT has two constraints:

1. All sentences must be padded or truncated to a single, fixed length.
2. The maximum sentence length is 512 tokens.

Padding is done with a special [PAD] token, which is at index 0 in the BERT vocabulary.

Training & validation split

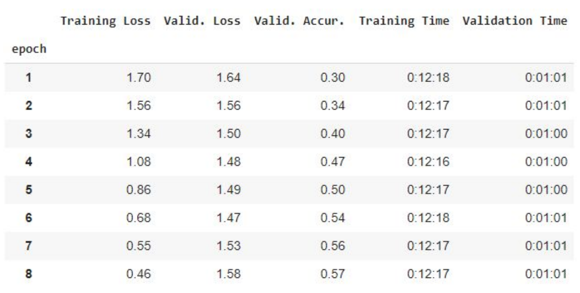
Next, We divide up our training set to use 80% for training and 20% for validation. We'll also create an iterator for our dataset using the torch DataLoader class to save on memory during training because, unlike a for loop, with an iterator the entire dataset does not need to be loaded into memory.

Training our classification model

We'll be using the BertForSequenceClassification model. This is the normal BERT model with an added single linear layer on top for classification that we will use as a sentence classifier. As we feed input data, the entire pre-trained BERT model and the additional untrained classification layer is trained on our specific task.

Performance & Analysis

We could see that the validation loss has consistently decreased until the 4th epoch and yielded an accuracy of 0.47 which when compared to 0.27 when trained without performing feature engineering, distillation and amalgamation was proven to be less efficient..



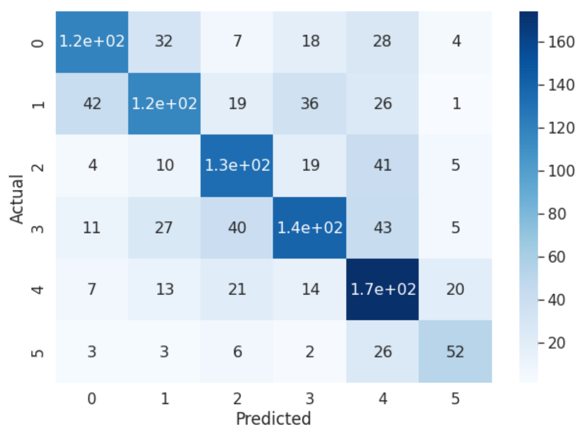
epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	1.70	1.64	0.30	0:12:18	0:01:01
2	1.56	1.56	0.34	0:12:17	0:01:01
3	1.34	1.50	0.40	0:12:17	0:01:00
4	1.08	1.48	0.47	0:12:16	0:01:00
5	0.86	1.49	0.50	0:12:17	0:01:00
6	0.68	1.47	0.54	0:12:18	0:01:01
7	0.55	1.53	0.56	0:12:17	0:01:01
8	0.46	1.58	0.57	0:12:17	0:01:01

Figure 3. Validation and Accuracies across eight training epochs

What did not work?

When we started off, we began with kaggle's fake news dataset. But the labels in this dataset as "fake" and "real" which does not really help in training the multi-class classification model.When we tried BERT multi-class classification with liar-liar dataset, although we have outperformed the baseline implementation of Naive Bayes model's accuracy, we did not achieve a good prediction and the evaluation resulted in overfitting after few epochs. This was because no other features were added to the encoded data to help the classification of the truth values. Without any data enrichment,amalgamation and distillation the performance of the model was not at its best and the model has not been used to its full potential.

Confusion matrix

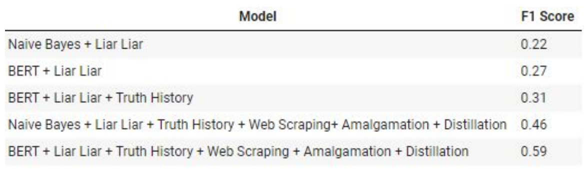


Actual \ Predicted	0	1	2	3	4	5
0	1.2e+02	32	7	18	28	4
1	42	1.2e+02	19	36	26	1
2	4	10	1.3e+02	19	41	5
3	11	27	40	1.4e+02	43	5
4	7	13	21	14	1.7e+02	20
5	3	3	6	2	26	52

Figure 4. Visualization of confusion matrix

What alternatives did you try?

First, along with the text, we have added features like the truth history counts of the speaker that are in the columns 9-13 which are encoded and passed into the classification layer for better prediction. This has shown an increase of 15% accuracy. Then, we started with data enrichment by scraping data from over 800 pages of politifact.com.Then we performed amalgamation with the liar-liar dataset. This drastically increased the F1 score from 0.28 to 0.59.



Model	F1 Score
Naive Bayes + Liar Liar	0.22
BERT + Liar Liar	0.27
BERT + Liar Liar + Truth History	0.31
Naive Bayes + Liar Liar + Truth History + Web Scraping+ Amalgamation + Distillation	0.46
BERT + Liar Liar + Truth History + Web Scraping + Amalgamation + Distillation	0.59

Figure 5. Comparison of F1 Scores

What did you research, what references (eg code) did you study or leverage (code)

I have researched various articles on data cleaning[7], topic modeling[8] and for code I have referred to the documentation of hugging face transformers for implementation details of BERT tokenizer[9] and BERT for sequence classification[10].

For multiclass classification using BERT I have referred to the code by mccormick written in his blog and provided the reference below[11].

B. Spam Classification

Dataset(s), Scraping, Enrichment

We have used three different datasets to classify any given information as spam or ham.

***Liar dataset*:** The liar dataset (Wang, 2017) large dataset collected through reliable annotation, it contains only short statements with its truth labels determined. This website collects statements made by US ‘speakers’ and assigns a truth value to them ranging from ‘True’ to ‘Pants on Fire’. The data contained news between 2016 and 2017 and is obtained by scraping from a politifact website.

***Fake news dataset*:** The latest hot topic in the news is fake news and many are wondering what data scientists can do to detect it and stymie its viral spread. It has a serious impact on our online as well as offline discourse. One can even go as far as saying that, to date, fake news poses a clear and present danger to western democracy and stability of the society.

This fake news dataset (Hassan Amin, 2019) is only a first step in understanding and tackling this problem. As per our requirement, this dataset is converted into spam or ham based on its veracity labels and is used for enrichment.

***Politifact scrapped dataset*:** This dataset is obtained by scraping from a politifact website. The data contained news information which is examined by human experts and determines the truthfulness of it. The labels determined vary from ‘true’, ‘mostly-true’, ‘half-true’, ‘barely-true’, ‘false’, ‘pants-fire’. The data is scraped and is then classified as spam or ham based on its truth labels and is amalgamated with the original dataset.

What did you try and what worked

Steps followed

1. Exploratory Data Analysis

Exploratory Data Analysis is a very important process of data science that refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies,to test hypotheses and to check assumptions with the help of summary statistics and graphical representations. It helps us understand the data at hand and relates it with the business context.

Word Cloud is a data visualization tool used for representing text data. The size of the texts in the image represent the frequency or importance of the words in the training data.

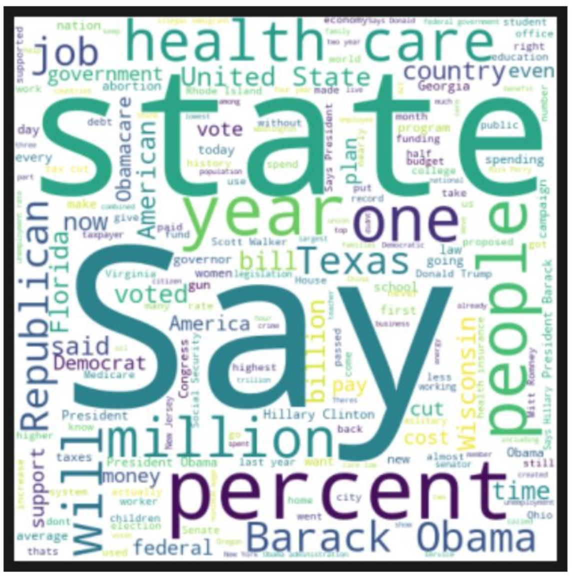


Figure 6. Word cloud for train.csv of liar dataset

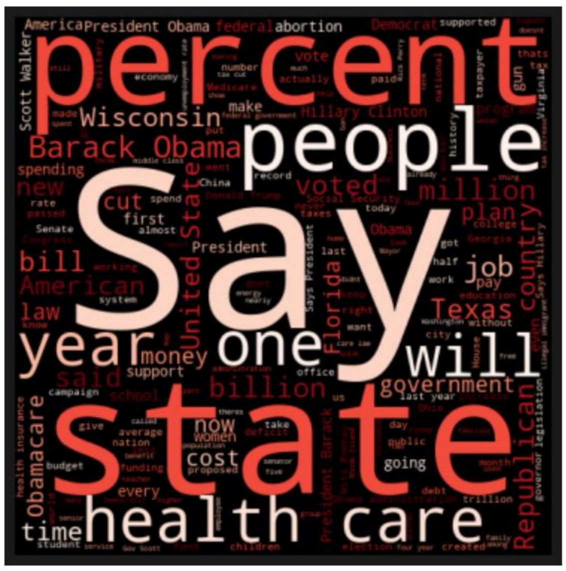


Figure 7. Word cloud for fake news dataset

```
Text(0.5, 0, 'Message Length')
```

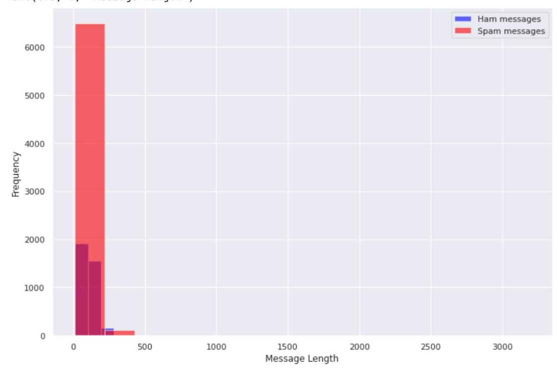


Figure 8. Representation of the trend of spam and ham texts in the liar dataset used in this paper

LDA is typically evaluated by either measuring performance on some secondary task, such as document classification or information retrieval, or by estimating the probability of unseen held-out documents given some training documents.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f88c4899fd0>
```

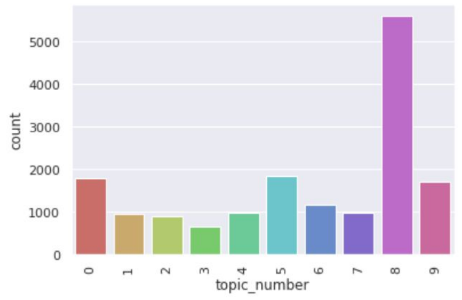


Figure 9. LDA modelling

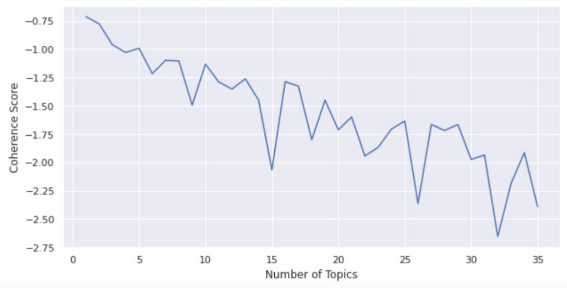


Figure 10. Graph represents Coherence score and number of topics obtained in LDA modelling

2. Pre processing

There are three main preprocessing techniques that are considered in training this model, they include removing all punctuation, removing all stopwords and returning a list of the cleaned text.

***Removing Stop Words*:** When using Natural Language Processing(NLP), our goal is to perform some analysis to remove stop words.

***Word Stemming*:** Words are reduced to their stemmed from removing the end or the beginning of the words, using a list of common prefixes and suffixes that can be found in that language. It usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

***Word Lemmatization**:* Lemmatization is utilizing the dictionary of a particular language and tries to convert the words back to its base form. It usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma* . If confronted with the token *saw*, stemming might return just *s*, whereas lemmatization would attempt to return either *see* or *saw* depending on whether the use of the token was as a verb or a noun.

3. Feature Extraction:

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. This paper contains the model that uses tfidf vectorizer as feature extraction technique.

Tf-idf vectorizer: In this process, the vectorizer will find words that often appear across multiple documents or sentences and try to downscale them. The approach that I took was to use the TfidfVectorizer as a feature extraction tool.

4. Applying algorithms

Nearest Neighbors: k-Nearest Neighbor is a lazy learning algorithm which stores all instances corresponding to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors)and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors.

Linear SVM: Support vector machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible.

Decision Tree: Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Random Forest: Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Neural Networks: A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

AdaBoost classifier: An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. It focuses on classification problems and aims to convert a set of weak classifiers into a strong one.

Logistic regression: Logistic Regression was used in the biological sciences in the early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

Naive Bayes: Naive Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. Naive Bayes classifiers work well in many real-world situations such as document classification and spam filtering

XGBoost: XGBoost is an optimized distributed gradient boosting that implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting also known as GBDT, GBM that solves many data science problems in a fast and accurate way.

RBF SVM: Support vector machines so called as SVM is a supervised learning algorithm which can be used for classification and regression problems as support vector classification (SVC) and support vector regression (SVR). Kernels in SVM classification refer to the function that is responsible for defining the decision boundaries between the classes. A RBF (radial basis function) kernel is used as a transformer or processor to generate new features by measuring the distance between all other dots to a specific dot/dots — center. RBF kernel uses two main parameters, gamma and C that are related to the decision region (how spread the region is), and the penalty for misclassifying a data point

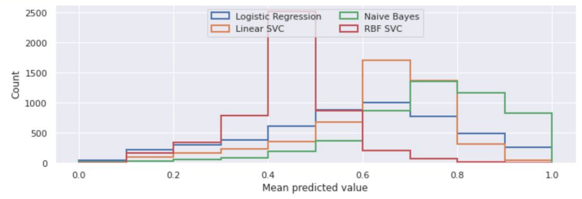


Figure 11. Mean predicted value vs count of the top 4 classifiers

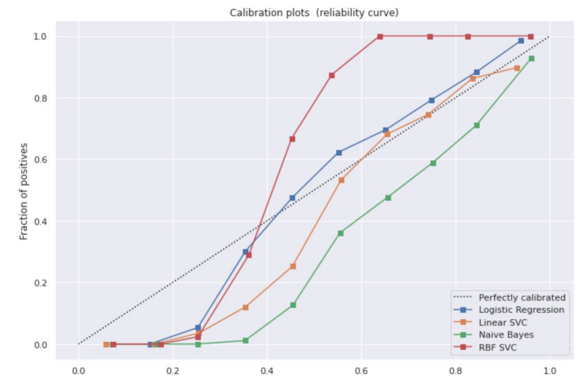
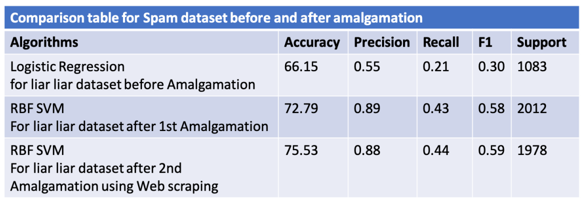


Figure 12. Calibration plots vs fraction of positives of the top 4 classifiers

5. Evaluation metrics

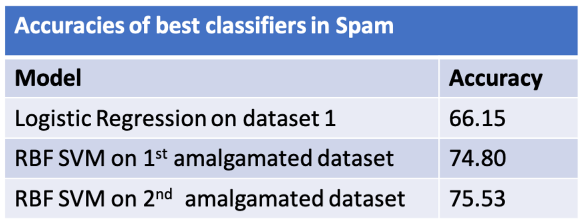
Evaluation metrics are used to measure the quality of the statistical or machine learning model. Evaluating machine learning models or algorithms is essential for any project. There are many different types of evaluation metrics available to test a model. These include classification accuracy, logarithmic loss, confusion matrix, and others.



Comparison table for Spam dataset before and after amalgamation					
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1**	**Support**
Logistic Regression for liar liar dataset before Amalgamation	66.15	0.55	0.21	0.30	1083
RBF SVM For liar liar dataset after 1st Amalgamation	72.79	0.89	0.43	0.58	2012
RBF SVM For liar liar dataset after 2nd Amalgamation using Web scraping	75.53	0.88	0.44	0.59	1978

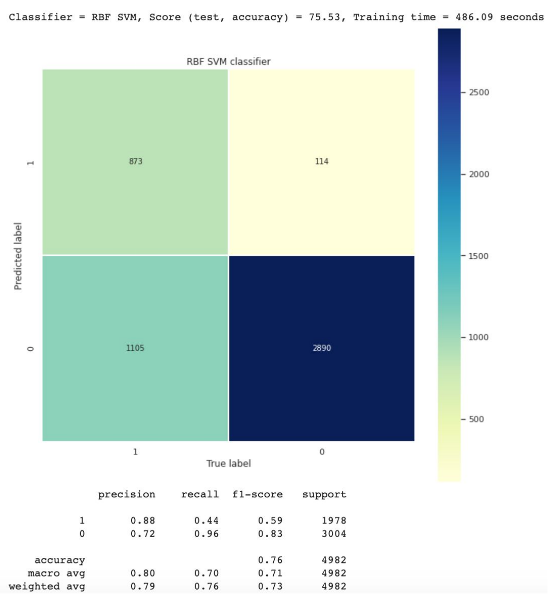
Figure 13. Comparison table for spam dataset before and after amalgamation for the best classifiers

A confusion matrix gives us a matrix as output and describes the complete performance of the model. There are other evaluation metrics that can be used that have not been listed. Evaluation metrics involve using a combination of these individual evaluation metrics to test a model or algorithm.



Accuracies of best classifiers in Spam	
Model	**Accuracy**
Logistic Regression on dataset 1	66.15
RBF SVM on 1st amalgamated dataset	74.80
RBF SVM on 2nd amalgamated dataset	75.53

Figure 14. Accuracy comparisons



```
Classifier = RBF SVM, Score (test, accuracy) = 75.53, Training time = 486.09 seconds
```

```
              precision    recall  f1-score   support

           1       0.88      0.44      0.59      1978
           0       0.72      0.96      0.83      3004

    accuracy                           0.76      4982
   macro avg       0.80      0.70      0.71      4982
weighted avg       0.79      0.76      0.73      4982
```

Figure 15. Confusion matrix after 2nd enrichment

The accuracy comparison of classifiers and results of accuracy graphs is plotted below that are obtained after enriching the liar dataset with fake news and politifact scraped data.

What did not work

Initially tried to implement the spam classifier on a spam dataset present in kaggle. But when the muller loop is applied, the highest accuracy was very low. And to improve accuracy, the implementation of LDA topic modelling on the liar dataset didn't give desirable results.

What alternatives did I try

At first, the model was trained on the kaggle spam dataset, it resulted in low accuracy. And then tried with the liar dataset by considering the labels and converting them into spam or ham. The highest accuracy obtained was 66.15% for logistic regression and to enrich the dataset, it is amalgamated with kaggle fake news dataset and then the accuracy has increased by almost near to 8.65%. To further increase the accuracy, another dataset is scraped from the politifact website and various classifiers are applied. The highest accuracy of the whole dataset is 75.53%. Thus the training was done on the model with this accuracy. This model is pickled and saved to further use to predict any given text information.

What did I research, what references (eg code) did I study or leverage (code)

There are different possible approaches to train a model. The references I followed for the model mentioned are as follows:

https://towardsdatascience.com/email-spam-detection-1-2-b0e06a5c0472

https://medium.com/analytics-vidhya/building-a-spam-filter-from-scratch-using-machine-learning-fc58b178ea56

https://towardsdatascience.com/spam-classifier-in-python-from-scratch-27a98ddd8e73

https://towardsdatascience.com/spam-or-ham-introduction-to-natural-language-processing-part-2-a0093185aebd

C. Stance Detection

Dataset(s), Scraping, Enrichment

We have used three different datasets for training various models to identify the stance of a news article.

Dataset 1: The SemEval 2016 Stance Detection for Twitter consists of tweets and targets for labelled training. The targets mainly include Climate Change is a Real Concern, Feminist Movement, Atheism, Legalization of Abortion and Hillary Clinton.

Dataset 2: Liar liar dataset has more than 10,000 articles already labeled for its veracity. They used a range of veracity which included 'true', 'mostly-true', 'half-true', 'barely-true', 'false', 'pants-fire'. Each article in this dataset contains details: label, statement, subject, speaker, speaker_job_title, state_info, party_affiliation, barely_true_counts,false_counts,half_true-counts , mostly_true_counts, pants_on_fire_counts, context.

Enrichment : Kaggle stance detection dataset. This dataset contains news articles with stances. The columns include Body ID, articleBody, Headline and Stance.

Scrapping: We have scrapped various new articles from Politifact.com along with the veracity labels. This data was used for inference and stance of these articles was identified using the model which gave best accuracy.

What did I try and what worked?

Steps followed:

- ***Pre-processing***

The text cleaning process begins with converting all the textual data to lowercase, followed by removal of stop words using NLTK library. The punctuations are also removed using the same library. Applied stemming, which is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Snowball stemmer is a language written to extract meaningful word roots from all forms of a word. Post stemming the text length is reduced by removing words whose length is less than Lemmatization is also performed on the data to return the dictionary form of a word. These preprocessing techniques are performed on both the train and test datasets.

- ***Visualization***

The pre-processed text is visualized to identify the average text length. This helps in deciding the vector size. Matlibplot library was used for visualization. The text categories are also visualized to understand the distribution of various classes of text.

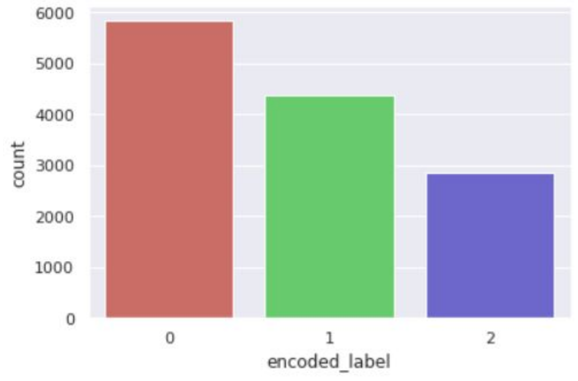


Figure 16. Data distribution for train dataset

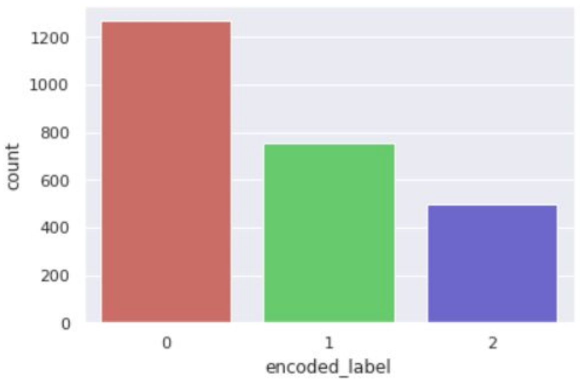


Figure 17. Data distribution for test dataset

Word-cloud is used to visualize the dataset. It provides more information about texts by analyzing correlations and similarities between words rather than analyzing texts only by the frequency of words appearing.

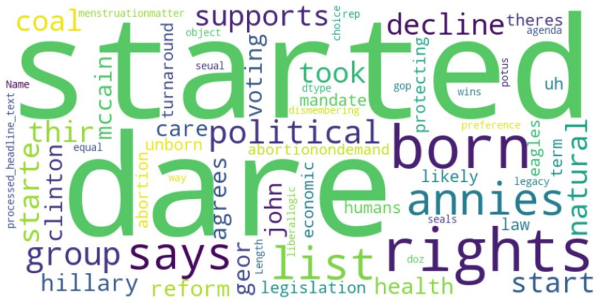


Figure 18. Word cloud visualization for train dataset

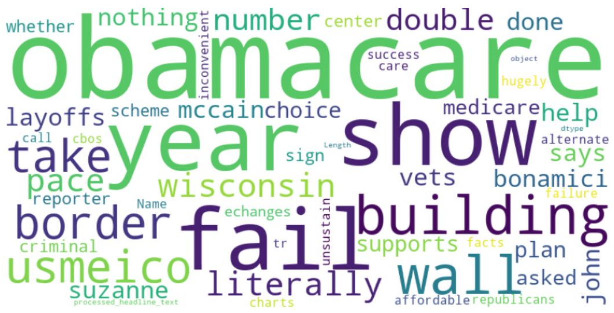


Figure 19. Word cloud visualization for test dataset

- ***LDA topic modelling***

The amalgamated dataset consists of tweets and news articles. To classify these articles into meaningful topics,we performed LDA on the text after performing the cleaning process in the dataset. Performed LDA predicting the top 10 topics and classifies the news articles based on these top 10 topics. This process also helped to categorize the news articles under top 10 topics.

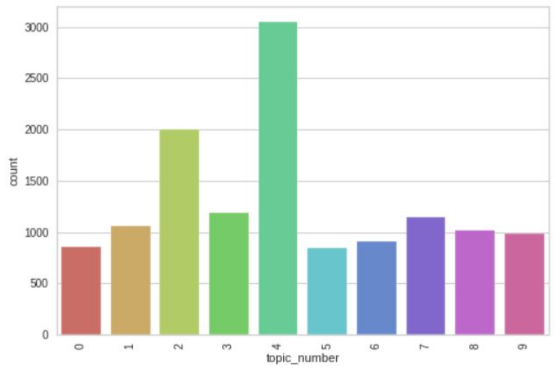
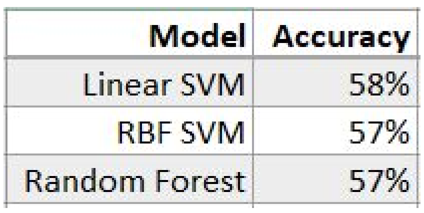


Figure 20. LDA Top 10 topic distribution

- ***Word2Vec***

Word2vec model is used to convert each word in a given text to a vector. We applied the Word2Vec model on the tweet dataset. This vector representation and targets were fed to various algorithms in the Muller loop. Linear SVM, RBF SVM and XGBoost classifiers have the highest accuracy.

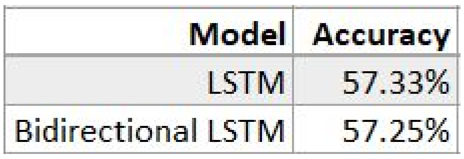


Model	Accuracy
Linear SVM	58%
RBF SVM	57%
Random Forest	57%

Figure 21. Word2Vec accuracy comparison

- ***LSTM approach***

The LSTM approach of stance detection shows the percentage of tweets which are in favor, against or neutral about the subject. One disadvantage from the LSTM is that it does not sufficiently take into account post word information because the sentence is read-only in one direction, ie, forward direction. To resolve this problem, we tried bidirectional LSTM, that is two LSTMs whose outputs accumulated together. First LSTM reads the sentence forward, and the second LSTM learns it backward. We concatenate the hidden states of each LSTM after they processed their own final word which gives a vector, which is fed to a fully connected hidden layer of size 30, and then passed through a softmax layer to provide the final classification probabilities. We also used dropout to reduce overfitting.

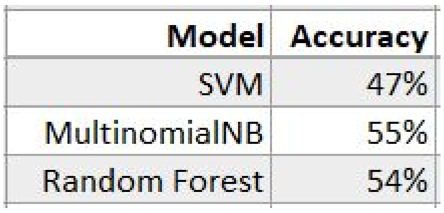


Model	Accuracy
LSTM	57.33%
Bidirectional LSTM	57.25%

Figure 22. LSTM model accuracy comparison

- ***Count Vectorization***

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. The scikit- learn CountVectorizer is used to convert text to vectors. The library takes a vocabulary list against which the frequency of occurrence is performed to convert text to vector. We amalgamated SemEval and Liar Liar datasets and used applied CountVectorizer. We used the Naives Bayes, Linear SVM and Random Forest classifiers.



Model	Accuracy
SVM	47%
MultinomialNB	55%
Random Forest	54%

Figure 23. Model accuracies using count vectorization

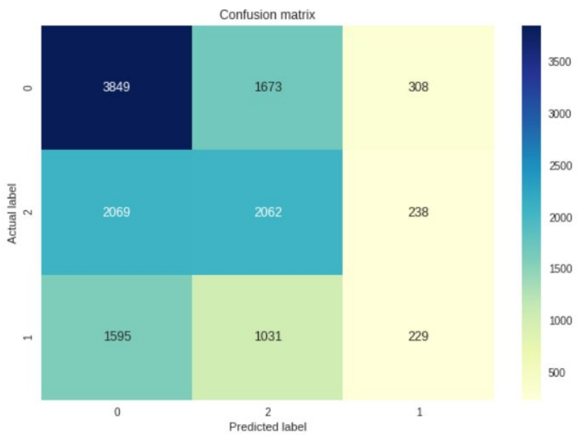
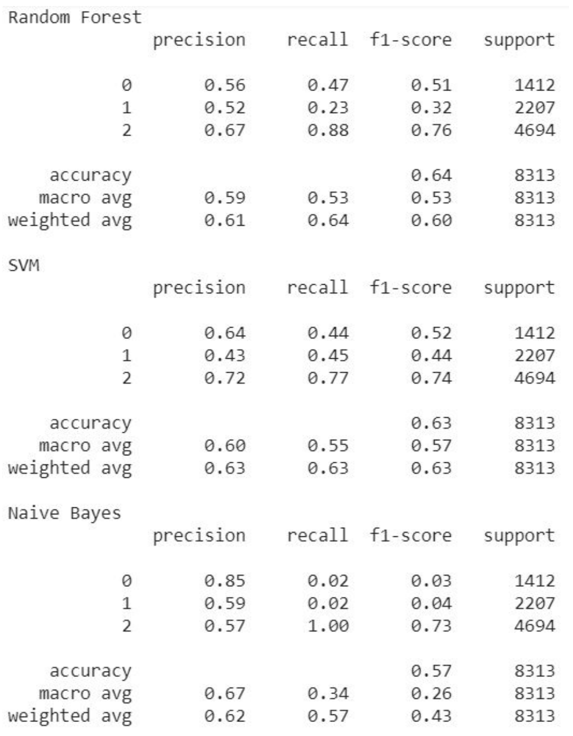


Figure 24. Confusion matrix for MultinomialNB

- **NGrams - TF-IDF Vectorization**

TF-IDF stands for term frequency-inverse document frequency. TF-IDF Vectorizer converts a collection of raw documents to a matrix of features. The approach was done to see if the accuracy scores would improve. We also amalgamated the Kaggle stance detection dataset to Twitter SemEval dataset and Liar Liar dataset. We used the Naives Bayes, Linear SVM and Random Forest classifiers on the amalgamated dataset and found that there was significant increase in the accuracy.



```
Random Forest
              precision    recall  f1-score   support

           0       0.56      0.47      0.51      1412
           1       0.52      0.23      0.32      2207
           2       0.67      0.88      0.76      4694

    accuracy                           0.64      8313
   macro avg       0.59      0.53      0.53      8313
weighted avg       0.61      0.64      0.60      8313

SVM
              precision    recall  f1-score   support

           0       0.64      0.44      0.52      1412
           1       0.43      0.45      0.44      2207
           2       0.72      0.77      0.74      4694

    accuracy                           0.63      8313
   macro avg       0.60      0.55      0.57      8313
weighted avg       0.63      0.63      0.63      8313

Naive Bayes
              precision    recall  f1-score   support

           0       0.85      0.02      0.03      1412
           1       0.59      0.02      0.04      2207
           2       0.57      1.00      0.73      4694

    accuracy                           0.57      8313
   macro avg       0.67      0.34      0.26      8313
weighted avg       0.62      0.57      0.43      8313
```

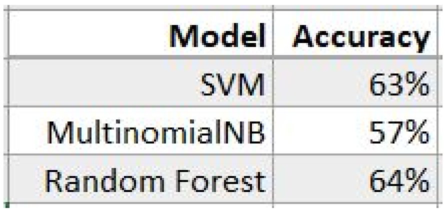
Figure 25. Report for models using TF IDF

What did not work?

The SemEval Task A dataset contained a limited tweet dataset. After amalgamating the dataset with the Liar liar dataset, there was a considerable decrease in accuracy.

What alternatives did you try?

We decided to amalgamate the Kaggle stance detection dataset with the combined dataset. The Kaggle dataset contains article body, headline and stance. We used the TF-IDF vectorizer technique and passed it to the top three classifiers. This improved the accuracy by 10%.



Model	Accuracy
SVM	63%
MultinomialNB	57%
Random Forest	64%

Figure 26. Final improved accuracies

What did I research, what references (eg code) did you study or leverage (code)

We have researched various articles on stance detection using various datasets like Fake News challenge dataset, SemEval dataset and RumorEval dataset. We checked the code available in the paper Stance Detection with Bidirectional Conditional Encoding and in the following location: https://github.com/sheffieldnlp/stance-conditional

We followed the approach mentioned in the paper and applied models with LSTM and Bidirectional LSTM layers and compared their accuracies with the standard baseline classifier models.

D. Clickbait

Dataset(s), Scraping, Enrichment

Dataset1:

LIAR LIAR- The LIAR dataset was published by William Yang in July 2017. He has retrieved the data from PolitiFact's API. This website collects statements made by US Speakers and assigns a truth value to them ranging from 'True' to 'Pants on Fire'.

Dataset2:

Kaggle Click bait dataset: The data is collected from various news sites. The clickbait headlines are collected from sites such as 'BuzzFeed', 'Upworthy', 'ViralNova', 'Thatscoop' and 'ViralStories . The non-clickbait headlines are collected from many trustworthy news sites such as 'WikiNews', 'New York Times', 'The Guardian', and 'The Hindu'.

Scraping:

Data scraped from different news websites using scrapy and News API.

What did you try and what worked ?

Steps followed:

Pre-processing:

- Tokenization of data:

The data is tokenized i.e., split into tokens which are the smallest or minimal meaningful units. The data is split into words.

- Converting to lowercase:

The data is converted into lowercase to avoid ambiguity between the same words in different cases like 'NLP', 'nlp' or 'Nlp'.

- Removing punctuation:

The punctuations are removed to increase the efficiency of the model. They are irrelevant because they provide no added information.

Removing Numbers: Removing numbers from the headlines.

- Removing Stop words:

Removing stop words using NLTK pandas library.

- Removing extra spaces:

Removing extra spaces with string operations.

- Lemmatization:

Lemmatization in linguistics is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form. It involves the morphological analysis of words. In lemmatization we find the root word or base form of the word rather than just clipping some characters from the end e.g. is, are, am are all converted to its base form in Lemmatization. Lemmatization is done using NLTK library.

Visualization:

Data visualization of Clickbait and Non-Clickbait.

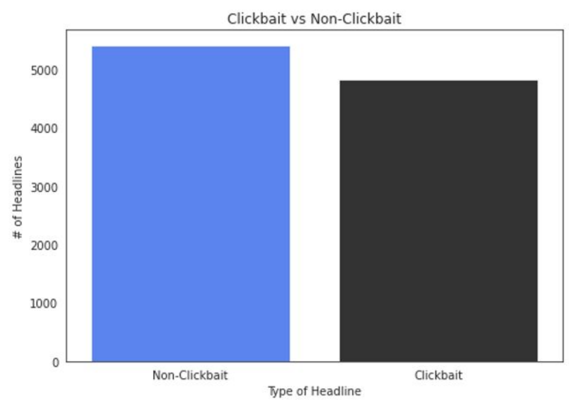


Figure 27. Data distribution of click bait and non-clickbait

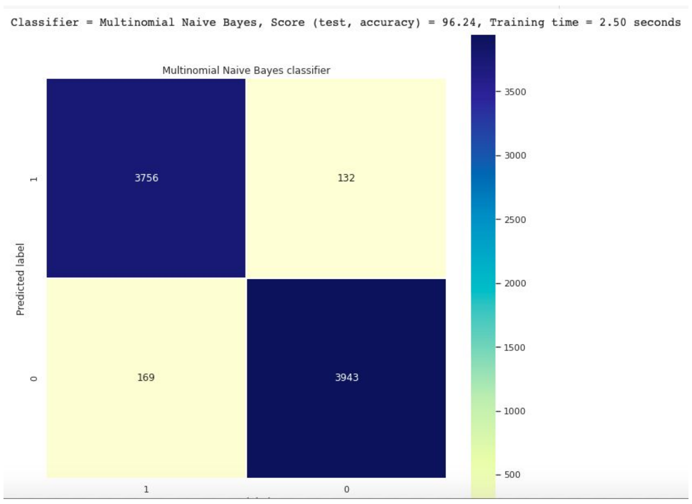
Feature Extraction:

TF-IDF (Term Frequency-Inverse Data Frequency): This method is used to convert the text into features.

Sentence Formality and Structure: To find out the sentence formality of the headlines we have used two features. The first one is F-Score and the second is the Coh-Matrix. F-score is calculated using part of speech tagging. Coh-Matrix is calculated by checking the sentence formation score.

Applying Classifiers:

Applied muller loop for all the preprocessed data and achieved 95% accuracy with naïve bayes classifier.



```
Classifier = Multinomial Naive Bayes, Score (test, accuracy) = 96.24, Training time = 2.50 seconds
```

Figure 28. Confusion matrix for Multinomial Naive Bayes

```
              precision    recall  f1-score   support

           1       0.97      0.96      0.96      3925
           0       0.96      0.97      0.96      4075

    accuracy                           0.96      8000
   macro avg       0.96      0.96      0.96      8000
weighted avg       0.96      0.96      0.96      8000
```

What did not work ?

We were not able to achieve good accuracy with liar liar dataset even after applying all the preprocessing techniques and distillation.

What alternatives did we try ?

We tried scraping and amalgamating the data with the liar dataset and increased accuracy from

50 percent to 96 percent for Naive Bayes classifier.

What did we research, what references (eg code) did we study or leverage (code) ?

Our research for clickbait detection was mainly around searching techniques for categorising the raw data like news headlines into clickbait and non-clickbait data. Below reference is used for achieving the same.

References:

https://arxiv.org/pdf/2003.12961.pdf

https://www.researchgate.net/publication/336888030_Clickbait_detection_A_literature_review_of_the_methods_used

IV. What steps did we take as a team?

We worked on the factors presented in the paper. We researched different techniques to preprocess and predict the accuracy. We used multiple approaches in order to extract features and use them in models. These works focus on textual content features. In addition, the different datasets have been amalgamated to make this prediction accurate. It can be seen that the main focus will be made on unsupervised and supervised learning models using textual news content. We also had common visualization methods and similar techniques for evaluating the classification model accuracy.

Polynomial equation approach: We made each of our factors into an importable model and imported each of them together to form a polynomial equation. The polynomial equation is derived based on the factors and the accuracy scores we received during classification and built a complete model for fake news classification. The procedure that we followed is as follows, firstly, we included the accuracy score from each of our factors and wrote a function to predict accuracy for any given text from each of our factors, and multiplied it with our accuracy.

Polynomial Equation = (0.96*Clickbait) + (0.64*Stance detection) + (0.76*Spam) + (0.47*Bert)

We classified the obtained score into different prediction labels also tested this by comparing it with scraped data thus resulting in an effective model

Multi-modal approach: In this method, we derived the individual embeddings for each of our factors - Clickbait detection, BERT multiclass classification, Stance detection, and Spam detection. We concatenated the embeddings from each factor and passed them through MLP, LSTM and Bidirectional LSTM approaches.A multilayer perceptron (MLP) is a deep, artificial neural network. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. We have given combined embeddings of all the factors through a sequence of dense and dropout layers which 'relu' activation function. Through a softmax layer to provide the final classification probabilities. We also tried the LSTM approach and achieved 50% accuracy. We also tried the Bidirectional LSTM approach where the initial LSTM reads the news article in forward direction, and the second LSTM learns it in a backward direction. In this approach, we are passing the input through a sequence of dense layers with relu activation function, dropout layers, Bidirectional LSTM and a final softmax layer. We have added early stopping with patience of 3. Achieved accuracy of 51%. We were able to classify any news into true, mostly true, half true, barely true, false, and pants on fire categories.

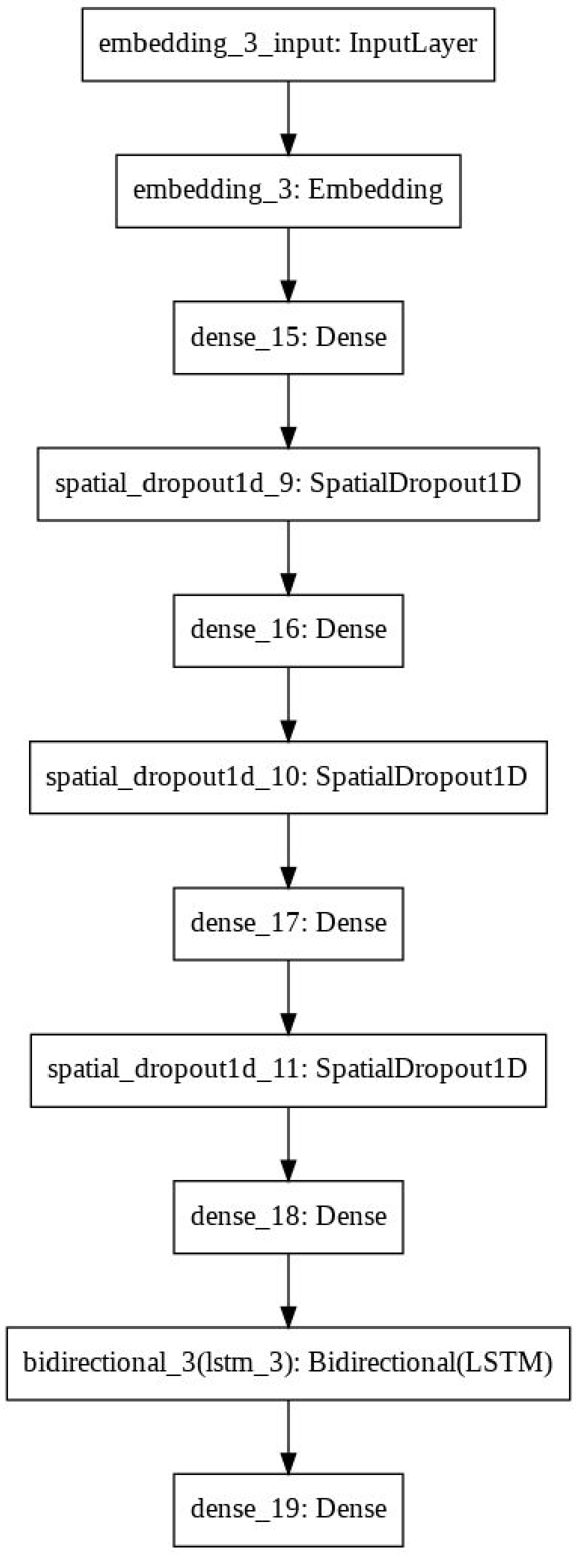

embedding_3_input: InputLayer
embedding_3: Embedding
dense_15: Dense
spatial_dropout1d_9: SpatialDropout1D
dense_16: Dense
spatial_dropout1d_10: SpatialDropout1D
dense_17: Dense
spatial_dropout1d_11: SpatialDropout1D
dense_18: Dense
bidirectional_3(lstm_3): Bidirectional(LSTM)
dense_19: Dense

Figure 28. Bidirectional LSTM architecture

Related work:

According to the FEVER paper[13], only headline text from Rashkin et al. dataset was used to train Naive Bayes and Support Vector Machine classifiers with TF-IDF features and various Convolutional Neural Network classifiers. These trained models were used to evaluate various datasets with different labels. Similarly we have used various models in Muller loop by passing TFIDF, Word2Vec and CountVectorizer for the datasets. But we have considered various factors like BERT multi-class classifier, Spam detection, Stance detection and Clickbait for determining the veracity of the news articles.

The paper considered the datasets separately and applied classifiers on each of them. The model did not achieve expected results as the training dataset did not have well-balanced data. We tried various techniques like amalgamation and distillation and considered factors to improve the performance of the model.

References:

1. "How To Design A Spam Filtering System with Machine Learning Algorithm", *Medium*, 2020. [Online]. Available:https://towardsdatascience.com/email-spam-detection-1-2-b0e06a5c0472.
2. "Redirecting", *Doi.org*, 2020. [Online]. Available: https://doi.org/10.1016/j.heliyon.2019.e0180
3. "Building a Spam Filter from Scratch Using Machine Learning", *Medium*, 2020. [Online]. Available: https://medium.com/analytics-vidhya/building-a-spam-filter-from-scratch-using-machine-learning-fc58b178ea56.
4. W. Awad and S. Elseuofi, "Machine Learning methods for E-mail Classification", International Journal of Computer Applications, vol. 16, no. 1, pp. 39-45, 2011. https://www.semanticscholar.org/paper/Machine-Learning-methods-for-E-mail-Clas

sification-Awad-Elseuofi/e0113ba1c6a737f685f84639645ce9b58b8ca41e

5. "Spam or Ham: Introduction to Natural Language Processing Part 2", *Medium*, 2020. [Online]. Available: https://towardsdatascience.com/spam-or-ham-introduction-to-natural-language-processing-part-2-a0093185aebd..
6. C. Conforti, M. T. Pilehvar, and N. Collier, "Towards Automatic Fake News Detection: Cross-Level Stance Detection in News Articles." https://www.aclweb.org/anthology/W18-5507.pdf
7. I. Augenstein and T. Rocktaschel, "Stance Detection with Bidirectional Conditional Encoding." https://www.aclweb.org/anthology/D16-1084.pdf
8. "NLP in Python-Data cleaning", Medium, 2020. [Online]. Available: https://towardsdatascience.com/nlp-in-python-data-cleaning-6313a404a470.
9. "Topic Modeling Tutorial with Latent Dirichlet Allocation (LDA)", Medium, 2020. [Online]. Available: https://towardsdatascience.com/topic-modeling-with-latent-dirichlet-allocation-by-example-3b22cd10c835.
10. "BERT — transformers 4.0.0 documentation", Huggingface.co, 2020. [Online]. Available: https://huggingface.co/transformers/model_doc/bert.html#berttokenizer.
11. "BERT — transformers 4.0.0 documentation", Huggingface.co, 2020. [Online]. Available: https://huggingface.co/transformers/model_doc/bert.html#bertforsequenceclassification.
12. "BERT Fine-Tuning Tutorial with PyTorch · Chris McCormick", Mccormickml.com, 2020. [Online]. Available: https://mccormickml.com/2019/07/22/BERT-fine-tuning/.
13. F. T. Asr and M. Taboada, "The Data Challenge in Misinformation Detection: Source Reputation vs. Content Veracity." https://www.aclweb.org/anthology/W18-5502.pdf